Midtveis

- Om midtveiseksamen
- Gjennomgang av prøveeksamen
- Kort oppsummering av pensum hittil

Midtveiseksamen 2025

- Tid: **17. mars kl 9:00–11:00**
- Sted: **Silurveien 2**
- Hjelpemidler: **Ingen!**
- Sensur: Bokstavkakterer **A–F**
- Sluttkarakter: Midtveiseksamen teller 25 %
  - Både midtveis- og avsluttende eksamen må bestås
  - De to må bestås samme semester

Det finnes en meget god nettside om eksamen i Silurveien:
https://www.uio.no/studier/eksamen/inspera/gjennomforing/

Oppmøte

Møt opp minst en halvtime før eksamen.

Trafikkproblemer er ingen unnskyldning.

Ikke på listen?

Hvis du møter opp og finner at du ikke står på listen selv om du mener du burde stått der, så snakk med en eksamensvakt. Du har rett til å gå opp til eksamen uansett, så blir saken avgjort i ettertid.

Varighet

Eksamen varer bare to timer.

- Det blir ikke tid for faglærerne til å gå «trøsterunde».

Men om du tror det er en feil i oppgaven?

1. Les oppgaveteksten én gang til.
2. Les oppgaveteksten enda en gang, og nøye.
3. Hvis du stadig mener at det er en *viktig* feil i oppgaven, kan du ta kontakt med eksamensvakten.

Automatrettet eksamen

Midtveiseksamen er laget for automatisk retting av Inspera.

- Følgende type oppgaver blir brukt:
  - Én- og flervalgsoppgaver
  - Dytte kodebiter på plass
  - Hva skjer?
  - ... og lignende

  Med andre ord: ingen tradisjonelle programmeringsoppgaver.
- Poeng: > 0 for rett svar, < 0 for galt svar, men aldri < 0 på en oppgave totalt.
- Sensuren blir klar dagen etter.
- Det er ikke mulig å klage på karakteren man får.

Ved sykdom

Hvis du er syk på eksamensdagen, må du ta kontakt med lege *samme dag* for å få en legeattest. Denne må leveres i et nettskjema innen tre dager.

Det arrangeres utsatt prøve («konteeksamen») i april for de som var syke og har levert legeattest.

Om du stryker

Du kan da melde deg til ny prøve («konteeksamen»).

Om du får en dårlig karakter

Trøst deg med at midtveiseksamen bare teller 25 %.

Det aller viktigste å ha med seg

- studentkortet
- UiO-brukernavn og passord

NB!

Sjekk at de fungerer på forhånd!

- blyant eller kulepenn

Midtveiseksamen 2024

Årets prøveeksamen var ordinær midtveiseksamen i 2024. Den er tilgjengelig i uio.inspera.no til og med lørdag.

702 studenter møtte. Resultatet ble:

A	29 %
B	28 %
C	34 %
D	6 %
E	2 %
F	1 %

Oppgave 1

Sett kodebitene på plass så det blir et lovlig og kjørbart program.

```
class Pakke {
   int vekt;
   Pakke (int v) { vekt = v; }
}

class LagPakke {

        }
}
```

din = min;

public static void main (String[] args) {

min = new Pakke(5);

int vekten = din.vekt;

Pakke min, din;

System.out.println(vekten);

Hvordan løse en slik oppgave?

1. Sett inn kode som *må* et gitt sted (f eks deklarasjonen av `main`-metoden).
2. Se så på deklarasjoner og bruk av variabler – dette gir sterke føringer på rekkefølgen.
3. Sjekk at programmet gjør det det skal.

Oppgave 2

Når er det naturlig å bruke arv og når er det naturlig med komposisjon i disse relasjonene?

	Arv	Komposisjon
Bok – illustrasjon	○	○
Hjul – sykkelhjul	○	○
Sete – sykkel	○	○
Bokhandel – bok	○	○
Kjøretøy – sykkel	○	○
Bil – taxi	○	○
Bil – hjul	○	○
Butikk – bokhandel	○	○
Terrengsykkel – sykkel	○	○
Person – professor	○	○

Hvordan løse en slik oppgave?

For hvert par av begreper still deg spørsmålet

- *Er A en del av B?*
  Da har vi en **komposisjon** (dvs en instansvariabel).
- *Er A en form for B?*
  Da har vi **arv** (dvs en subklasse).

Oppgave 3

Lag et program **MinstAv5** som leser fem ikke-negative heltall og skriver ut det minste.

```
import java.util.Scanner;
class [          ] {
  public static void main (String[ ] args) {
      Scanner tastatur = new Scanner(System.in);
      int [          ] ;
      for ( [          ] ) {
            [          ]
            if ( [          ] ) {
                  [          ] ;
            }
      }
      System.out.println( [          ] );
  }
}
```

v	MinstAv5	int v = tastatur.nextInt();	int i = 1; i < 5; i++	i = 5	min = v	
i == 1 || v < min	i	min	min == i	min = 0	int i = 1; i <= 5; i++	

Hvordan løse en slik oppgave?

- Som nevnt tidligere: Plasser først de kodebitene som er udiskutable.
- Les også oppgaveteksten nøye, for eksempel der det angis hva programmet skal hete. Da vet vi klassens navn.
- Sjekk spesielt kodebiter som er nesten like – forskjellen avgjør hvilken som skal velges.
- Bruk erfaringen du har opparbeidet som programmerer, for eksempel: Hva skjer om vi deklarerer `min` uten å initere den?

Oppgave 4

Hva skrives ut av dette programmet?

NB! Legg merke til at *tall2* er definert som **static**!

Eva: ☐ og ☐
Ole: ☐ og ☐

```
class StatiskPerson {
    public static void main (String[] args) {
        Person eva = new Person("Eva", 3);
        Person ole = new Person("Ole", 11);
        eva.oekTall(4);
        ole.oekTall(-1);
    }
}

class Person {
    int tall1;
    static int tall2;
    String navn;

    Person(String id, int n) { navn = id;  tall1 = n;  tall2 = 0; }

    void oekTall (int n) {
        tall1 = tall1 + n;
        tall2 = tall2 + (2 * n);
        System.out.print(navn + ": " + tall1);
        System.out.println(" og " + tall2);
    }
}
```

Hvordan løse en slik oppgave?

I slike «hva skjer»-oppgaver er det lurt å bruke tegninger:

	tall1	tall2
Eva:	3	0
Ole:	11	

	tall1	tall2
Eva:	7	8
Ole:	11	

	tall1	tall2
Eva:	7	6
Ole:	10	

Oppgave 5

Hvilke navn skrives ut?

	Eva	Ole	Per
Nr 1	O	O	O
Nr 2	O	O	O
Nr 3	O	O	O

```java
class Person {
    String navn;
    Person (String n) { navn = n; }
}

class LagEiere {
    public static void main (String[] args) {
        Person eva = new Person("Eva");
        Person ole = new Person("Ole");
        Person per = new Person("Per");
        Person meg, du, nabo, vennen, kollega;

        meg = eva;
        du = ole;
        nabo = per;
        kollega = du;
        vennen = eva;
        du = vennen;
        System.out.println("Nr 1: " + du.navn);
        System.out.println("Nr 2: " + meg.navn);
        System.out.println("Nr 3: " + kollega.navn);
    }
}
```

Hvordan løse en slik oppgave?

Igjen, ved slike «hva skjer»-oppgaver er det lurt å tegne litt:

`eva` → `"Eva"` `ole` → `"Ole"` `per` → `"Per"`

`meg` → `"Eva"` `du` → `"Ole"` `nabo` → `"Per"`

`kollega` → `"Ole"` `vennen` → `"Eva"` `du` → `"Eva"`

Oppgave 6

Hva skrives ut av dette programmet?

Tutsi veier: ☐

Mons veier: ☐

Gråpus veier: ☐

```
class KattOgMusOgOst {
    public static void main (String[] args) {
        Katt tutsi = new Katt(3000);
        Katt mons = new Katt(4000);
        Katt graapus = new Katt(5000);

        Mus pip = new Mus(200);
        Mus lille = new Mus(300);
        Mus smal = new Mus(400);

        Ostebit minst = new Ostebit(10);
        Ostebit liten = new Ostebit(20);
        Ostebit stor = new Ostebit(30);

        pip.spis(minst);
        lille.spis(liten);

        mons.spis(pip);
        tutsi.spis(lille);
        graapus.spis(smal);

        System.out.println("Tutsi veier: " + tutsi.vekt);
        System.out.println("Mons veier: " + mons.vekt);
        System.out.println("Gråpus veier: " + graapus.vekt);
    }
}

class Katt {
    int vekt;
    Katt (int v) { vekt = v; }
    void spis (Mus m) { vekt = vekt + m.vekt; }
}

class Mus {
    int vekt;
    Mus (int v) { vekt = v; }
    void spis (Ostebit ost) { vekt = vekt + ost.vekt; }
}

class Ostebit {
    int vekt;
    Ostebit (int v) { vekt = v; }
}
```

Hvordan løse en slik oppgave?

Det viktige i denne oppgaven er å holde oversikten over alles vekt:

Pip	200 ⇒	210
Lille	300 ⇒	320
Mons	4000 ⇒	4210
Tutsi	3000 ⇒	3320
Gråpus	5000 ⇒	5400

Oppgave 7

Hvilke tallverdier skrives ut av dette programmet?

Tall nr 0: ☐

Tall nr 1: ☐

Tall nr 2: ☐

Tall nr 3: ☐

```
class TestFrekvens {
    public static void main (String[] args) {
        Frekvens frek = new Frekvens(4);
        for (int i = 0;  i<=4;  i++) {
            frek.observert(i);
            frek.observert(i+1);
            frek.observert(i+2);
        }
        frek.observert(2);
        frek.observert(4);

        for (int ind = 0;  ind < frek.hentAntall();  ind++) {
            System.out.println("Tall nr " + ind + ": " + frek.hentVerdi(ind));
        }
    }
}

class Frekvens {
    private int[] tall;

    Frekvens (int ant) { tall = new int[ant]; }

    void observert (int indeks) {
        if (indeks >= 0 && indeks < tall.length)
            tall[indeks] = tall[indeks] + 1;
    }

    int hentAntall () { return tall.length; }
    int hentVerdi (int indeks) { return tall[indeks]; }
}
```

Hvordan løse en slik oppgave?

Vi skal telle hvor mange ganger et tall i et gitt intervall forekommer.
NB! Tallverdier utenfor intervallet blir ignorert.

	[0]	[1]	[2]	[3]
	0	0	0	0
i=0: 0, 1, 2	1	1	1	0
i=1: 1, 2, 3	1	2	2	1
i=2: 2, 3, 4	1	2	3	2
i=3: 3, 4, 5	1	2	3	3
i=4: 4, 5, 6	1	2	3	3
2:	1	2	4	3
4:	1	2	4	3

Oppgave 8: Rekursjon

I år blir rekursjon undervist først etter midtveiseksamen.

Oppgave 9

Dette programmet skriver ut 8 verdier.
Hvilke verdier skrives ut?

Bruttopris 0:	☐	Vanlig pris 0:	☐
Bruttopris 1:	☐	Vanlig pris 1:	☐
Bruttopris 2:	☐	Vanlig pris 2:	☐
Bruttopris 3:	☐	Vanlig pris 3:	☐

```
class Polymorfi {
    public static void main (String[] args) {
        Vare[] alleVarer = new Vare[4];
        alleVarer[0] = new Vare(3);
        alleVarer[1] = new Matvare(7);
        alleVarer[2] = new Nødmat(12);
        alleVarer[3] = new IkkeMat(20);

        for (int i = 0;  i < 4;  i++) {
            System.out.println("Bruttopris " + i + ": " +
                                alleVarer[i].bruttoPris());
            System.out.println("Vanlig pris " + i + ": " +
                                alleVarer[i].pris);
        }
    }
}

class Vare {
    int pris;
    Vare (int v) { pris = v; }
    public int bruttoPris () { return pris * 2; }
}

class Matvare extends Vare {
    Matvare (int v) { super(v); }
    @Override public int bruttoPris() { return pris; }
}

class Nødmat extends Matvare {
    Nødmat (int v) { super(v); }
    @Override public int bruttoPris() { return pris / 2; }
}

class IkkeMat extends Vare {
    IkkeMat (int v) { super(v); }
}
```

Hvordan løse en slik oppgave?

- Instansvariabelen `pris` endres ikke, så den kan leses rett ut.
- Ellers er det spørsmål om å finne den rette `bruttoPris`.

	B 0	P 0	B 1	P 1	B 2	P 2	B 3	P 3
`pris`		3		7		12		20
0: `Vare`: `Vare.bruttoPris()`	6	3		7		12		20
1: `Matvare`: `Matvare.bruttoPris()`	6	3	7	7		12		20
2: `Nødmat`: `Nødmat.bruttoPris()`	6	3	7	7	6	12		20
3: `IkkeMat`: `Vare.bruttoPris()`	6	3	7	7	6	12	40	20

Oppgave 10

Dette lille programmet tester tilordning og typekonvertering av referanser til objekter. Av de ti indikerte linjene, hvilke

- kompileres og kjøres uten problemer?
- vil gi kompileringsfeil eller feil under kjøring?

```
interface Egglegger {
    int antallEgg ();
}

class Hare {
    final String navn;
    Hare (String n) { navn = n; }
}

class Påskehare extends Hare implements Egglegger {
    int antall;
    Påskehare (String n, int ant) { super(n);  antall = ant; }

    @Override public int antallEgg () { return antall; }
}

class Høne implements Egglegger {
    @Override public int antallEgg () { return 1; }
}

class BrukHare {
    public static void main (String[] args) {
        Hare skogshare = new Hare("Bjørk");
        Hare sesonghare = new Påskehare("Påsan", 999);
        Høne burhøne = new Høne();

        // Hvilke av disse 10 linjene er lovlige som neste linje?
        Påskehare h1 = sesonghare;
        Egglegger h2 = sesonghare;
        Egglegger h3 = burhøne;
        Egglegger h4 = (Egglegger)skogshare;
        Påskehare h5 = (Påskehare)sesonghare;
        Egglegger h6 = (Hare)sesonghare;
        Hare      h7 = sesonghare;
        Egglegger h8 = (Egglegger)sesonghare;
        Egglegger h9 = skogshare;
        Påskehare h10 = (Påskehare)skogshare;
    }
}
```

Hvordan løse en slik oppgave?

Når er tilordning lov?

- `T a = new TT(...);`
  Er TT samme som T eller en implementasjon/subklasse av T?
- `T a = b;`
  Er b sin ***deklarasjon*** T eller en implementasjon/subklasse av T?
- `T a = (TT)b;`
  - Er b ***opprettet*** som et objekt av klassen TT eller en implementasjon/subklasse av TT?
  - Er TT klassen T eller en implementasjon/subklasse av T?

Hovedregel

Automatisk konvertering er lov når vi går til mer generelle (dvs «suprere») klasser.

- **`Påskehare h1 = sesonghare;` NEI**
  `sesonghare` deklarert som `Hare` som ikke er subklasse av `Påskehare`
- **`Egglegger h2 = sesonghare;` NEI**
  `Hare` implementerer ikke `Egglegger`
- **`Egglegger h3 = burhøne;` JA**
  `Høne` implementerer `Egglegger`
- **`Egglegger h4 = (Egglegger)skogshare;` NEI**
  `skogshare` opprettet med **`new Hare(...)`** og `Hare` implementerer ikke `Egglegger`
- **`Påskehare h5 = (Påskehare)sesonghare;` JA**
  `sesonghare` opprettet med **`new Påskehare(...)`**

- **`Egglegger h6 = (Hare)sesonghare;` NEI**
  `sesonghare` opprettet som `Hare`, men `Hare` implementerer ikke `Eggelgger`
- **`Hare h7 = sesonghare;` JA**
  `sesonghare` deklarert som `Hare`
- **`Egglegger h8 = (Egglegger)sesonghare;` JA**
  opprettet som `Påskehare` som implementerer `Egglegger`
- **`Egglegger h9 = skogshare;` NEI**
  deklarert som `Hare` som ikke implementerer `Egglegger`
- **`Påskehare h10 = (Påskehare)skogshare;` NEI**
  opprettet som `Hare` som ikke er subklasse av `Påskehare`

Hovedpunkter i pensum hittil

- Programmeringsspråket Java
  - Klasser og interface-er
    - Subklasser
    - Konstruktører
    - Polymorfi
    - Generiske klasser
  - Unntak
  - Iteratorer
- Modellering (dvs definere datastruktur for et problem)
  - Arv og komposisjon
  - Klassediagrammer
  - Datastrukturtegninger
- Beholdere
  - Arrayer
  - Lenkelister i ulike varianter

Sjekkliste for klasser og objekter

Objektorientert programmering dreier seg om å bruke objekter til å modellere/programmere ting eller begreper.

Klasser definerer hvordan et objekt skal representeres.

Subklasser definerer spesialiseringer av klasser.

Interface-er beskriver egenskaper som klasser kan implementere.

Statiske variabler og metoder finnes i «selve klassen».

Referanser peker på objekter; deklarasjonen angir ønsket subklassenivå.

Typekonvertering kan gi oss et annet «blikk» (dvs subklassenivå) på et objekt.

Polymorfe metoder velger riktig redefinert (`@Override`-et) metode.

Generiske klasser har en typeparameter (f eks `class Kø<T>`).

Bruk av klasser og objekter

Noen ting å huske på med klasser, subklasser og objekter:

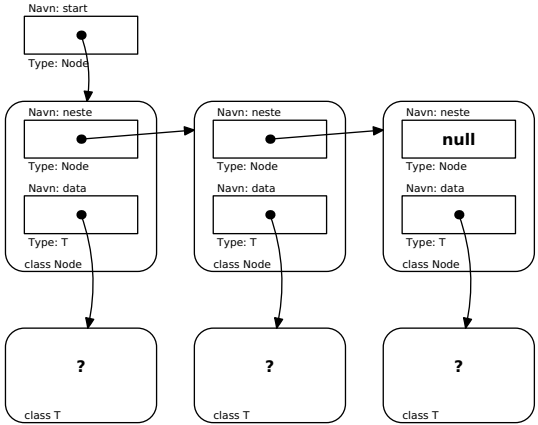
- Et objekt opprettet med **`new Xxx(...)`** er det alltid av den klassen.
- **`instanceof`** kan fortelle oss om objektet er av en gitt klasse.
- Når vi kaller på metoder, kalles alltid (re)definisjonen i selve objektet.

Bruk av referanser

- Alle referanser er deklarert til å være av en gitt type (dvs klasse/interface).
- Referanser kan angi objekter av den gitte typen eller en subklasse/implementasjon. Dette sjekkes av kompilatoren.
- Ved å bruke typekonvertering («casting») kan vi angi at vi ønsker et annet blikk på referansen. Kjøresystemet sjekker om dette er lov.

Lenkelister

En meget fleksibel datastruktur med objekter kjedet sammen med pekere.



Det finnes mange varianter:

Pekere i selve objektet istedenfor å ha egne node-objekter

 Sparer plass  Mindre generelle; objekter kan ikke være i flere lister

Sluttpeker i tillegg til startpekeren

 Litt mer komplisert kode  Mye raskere for operasjoner på slutten

Dobbeltlinket

 Mer komplisert kode  Mye raskere for endring ikke i en ende

Egne hode- og haleobjekter

 Tar litt plass  Endringer blir enklere og litt raskere

Forberedelse til eksamen

- Programmer så mye som mulig
  Forslag:
  - Hent kode fra forelesningene. Få den til å virke. Se hva som skjer om du endrer den.
  - Trix-oppgavene.
  - Oppgaver i læreboken *Big Java*.
  - Finn på oppgaver selv!
    - Lagre spillelisten din på strukturert form med diverse subklasser.
    - Lag en klasse `Brøk`, `Kortstokk`, `Dato`, ...
- Gjennomfør prøveeksamen og sjekk svarene med fasiten på https://www.uio.no/studier/emner/matnat/ifi/IN1010/v25/midtveiseksamen/midtveis-v24-fasit.pdf.